

# DeCaught

Monitor de ubicación

Un proyecto de:

Samuel Antequera  
Nil Mitjans  
Aitor Rodríguez



# Análisis

Descripción del servicio



# ¿Qué es DeCaught y a qué público está orientado?

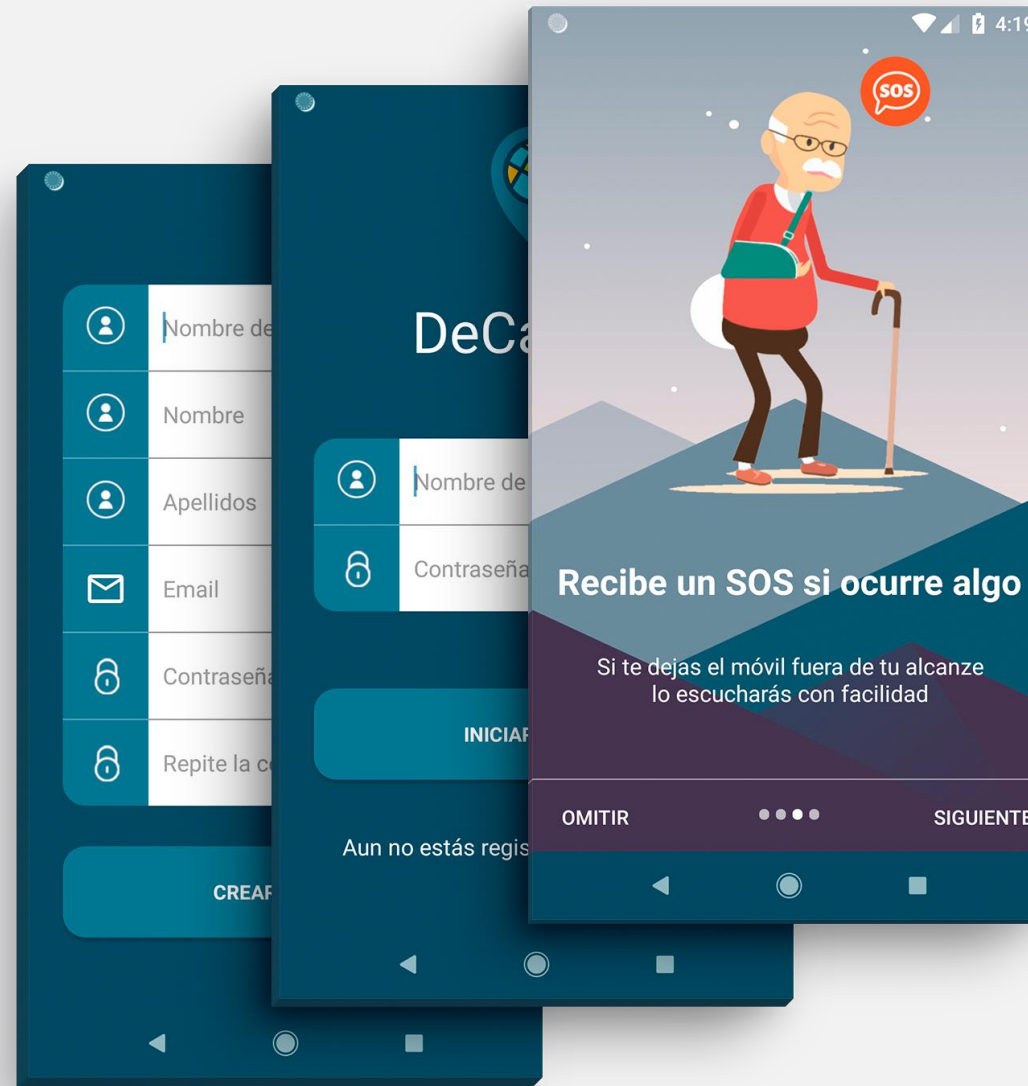
- DeCaught es un monitor de ubicación de dispositivos. Rastrea, almacena y monitoriza la ubicación de personas con problemas de memoria o episodios de desorientación.
- El sistema muestra la ubicación de la persona y alerta cuanto está fuera de las áreas delimitadas.
- Está destinado a las personas que por edad o enfermedad se pierden y/o desorientan de manera habitual.
- DeCaught está pensado para que puedan mantener cierta autonomía en su día a día.

Análisis

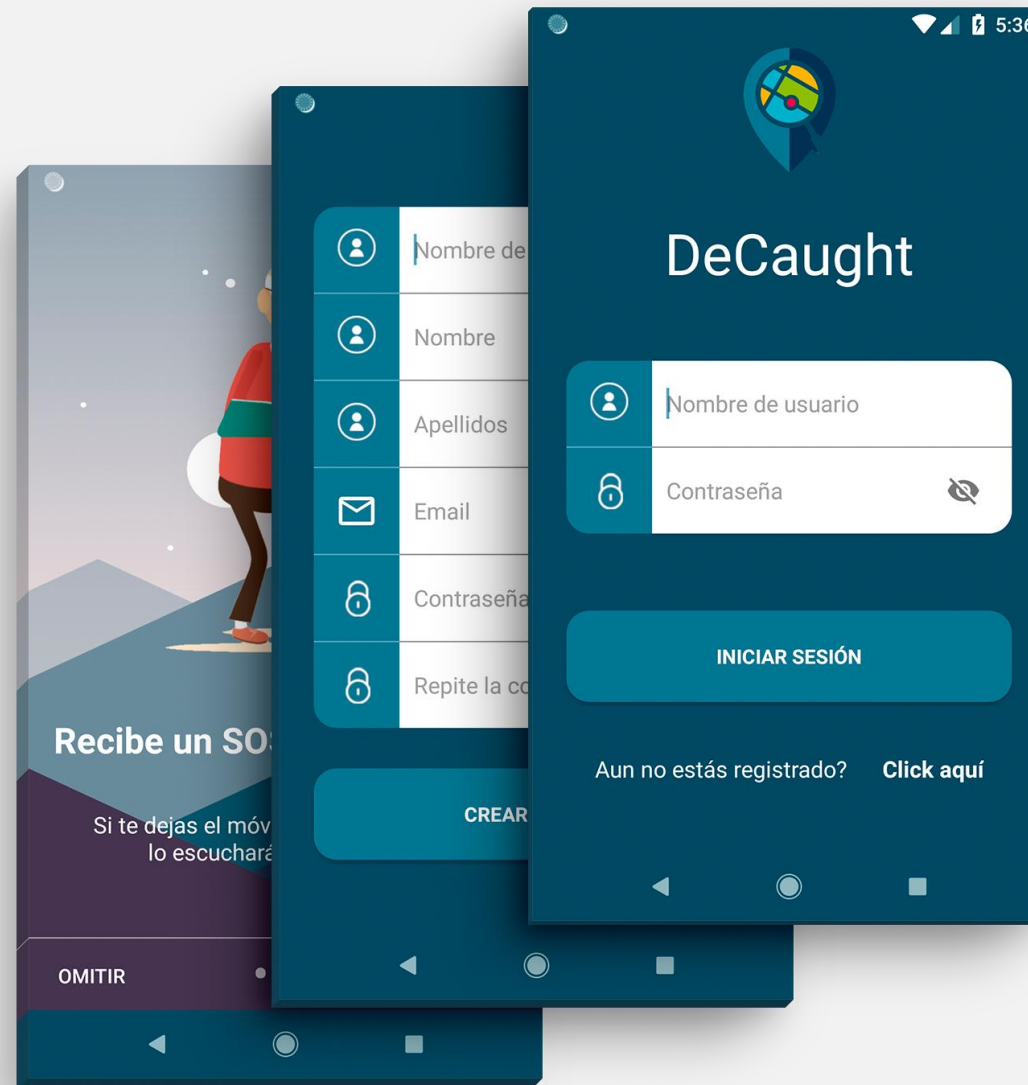
La aplicación



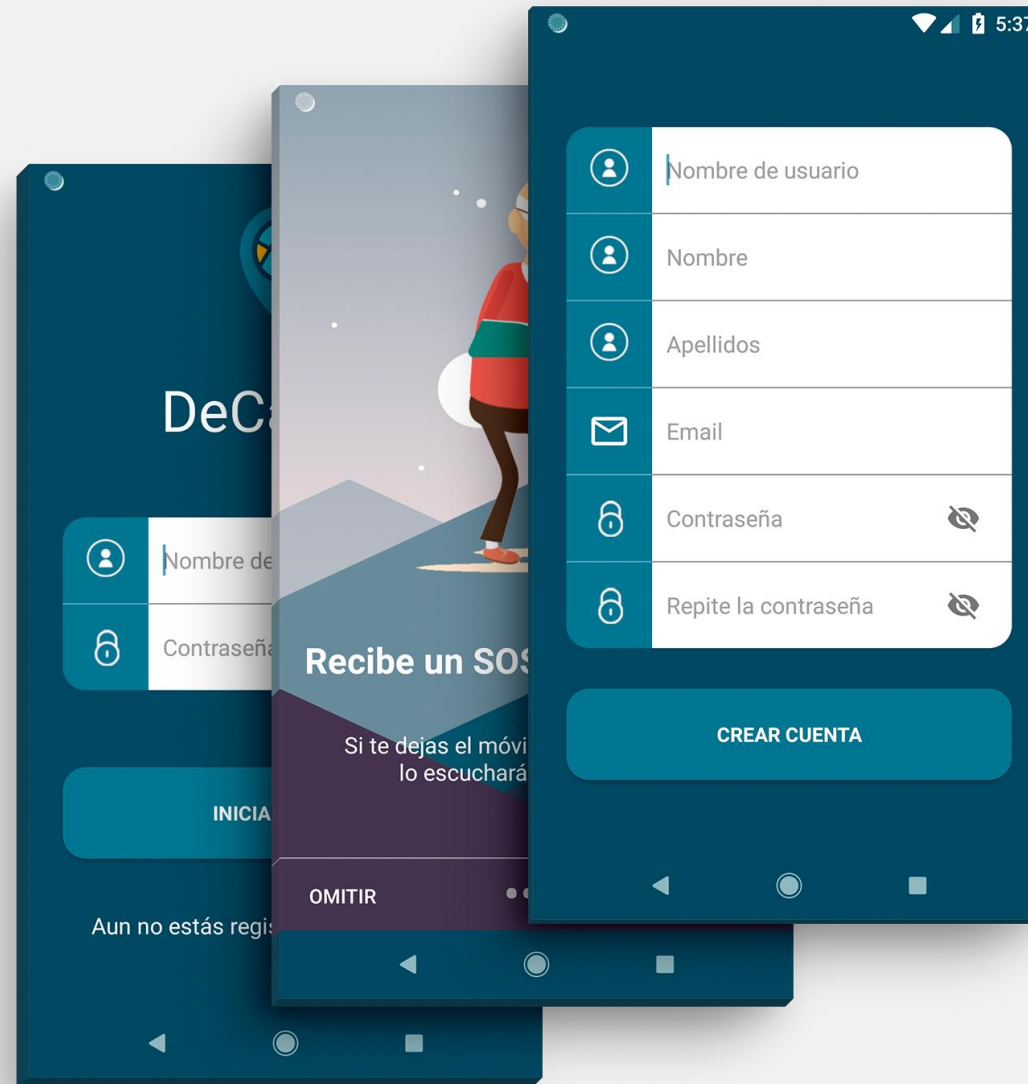
# Bienvenida, login y registro



# Bienvenida, login y registro



# Bienvenida, login y registro



# Selección de rol y registro del dispositivo

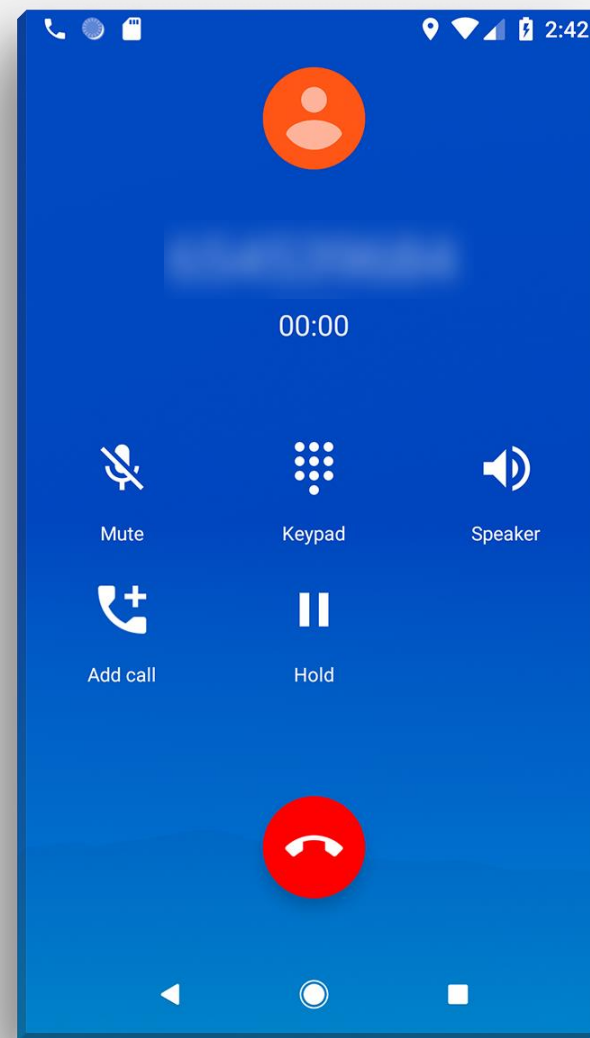
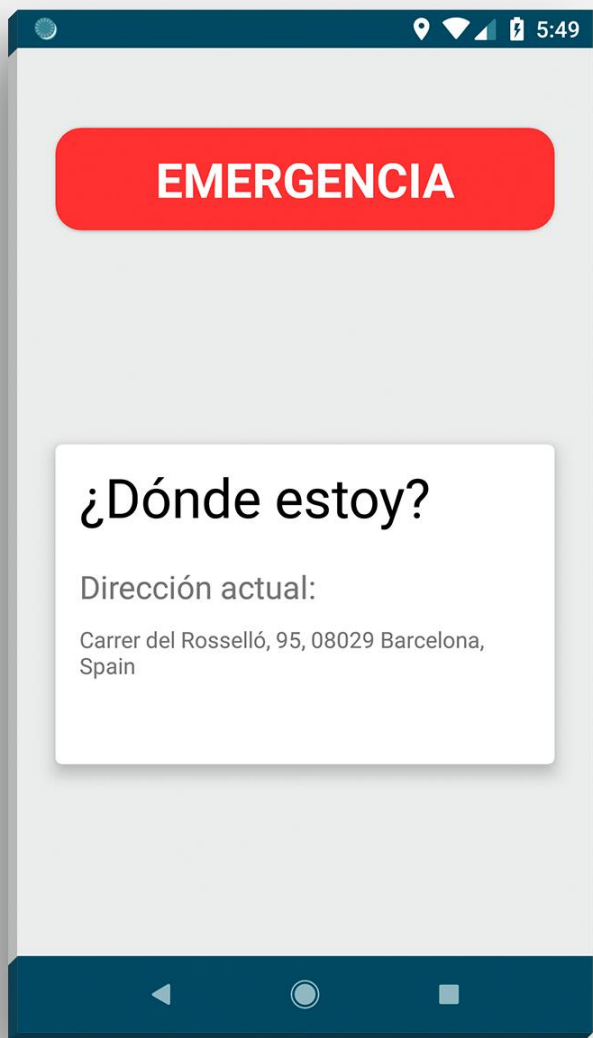




# Pantalla del receptor



# Pantalla del emisor y llamada de emergencia

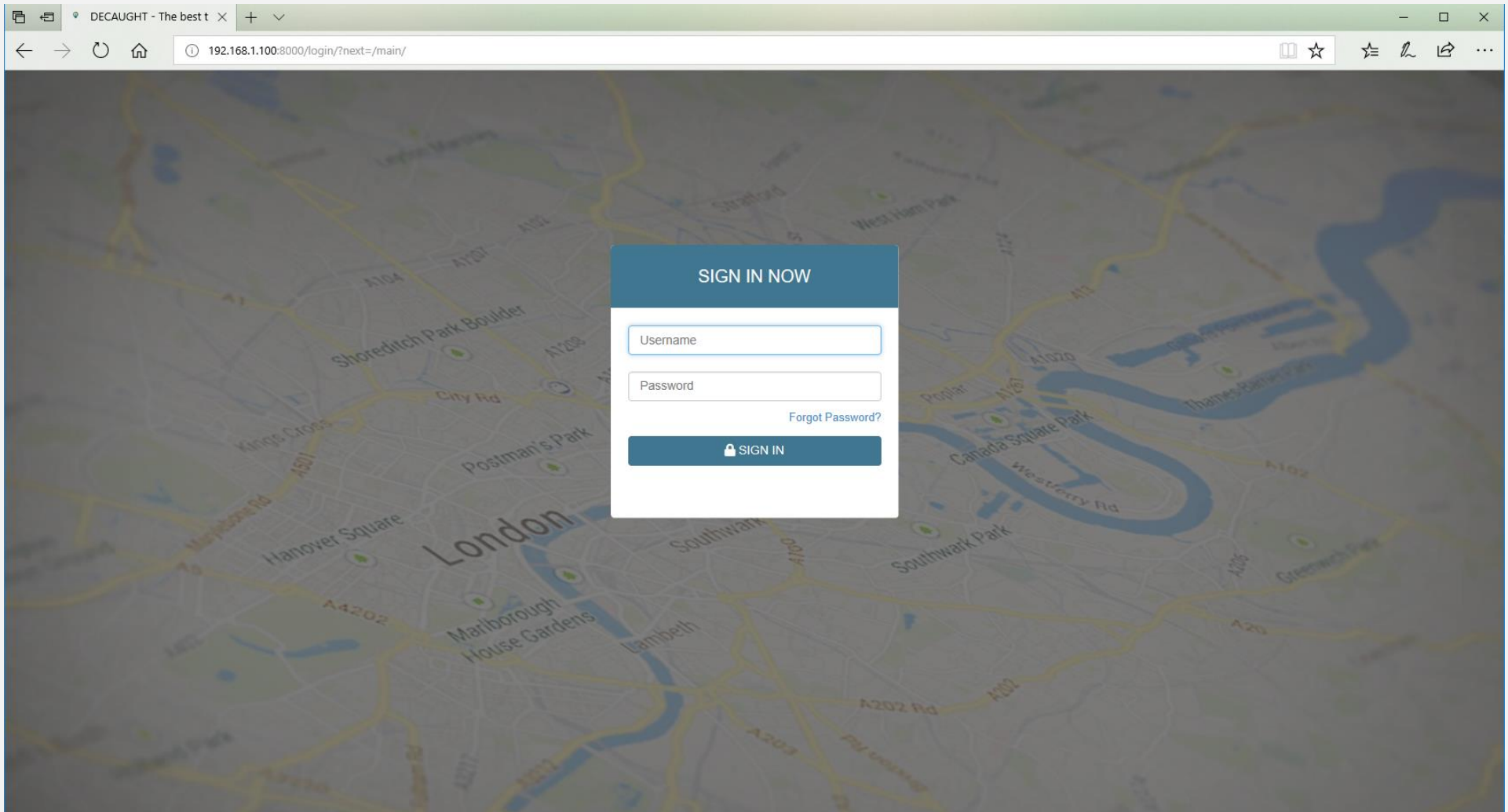


# Análisis

La página web



# Login



The screenshot shows a web browser window with a single tab titled "DECAUGHT - The best t". The address bar displays the URL "192.168.1.100:8000/login/?next=/main/". The browser's toolbar includes navigation buttons (back, forward, refresh, home), a star icon for bookmarks, and a menu icon. The main content area features a map of London as a background. Overlaid on the map is a white login form with a blue header bar that reads "SIGN IN NOW". The form contains two input fields: "Username" and "Password". Below the "Password" field is a blue link labeled "Forgot Password?". At the bottom of the form is a blue button with a white lock icon and the text "SIGN IN".

DECAUGHT - The best t


192.168.1.100:8000/login/?next=/main/

SIGN IN NOW

Username

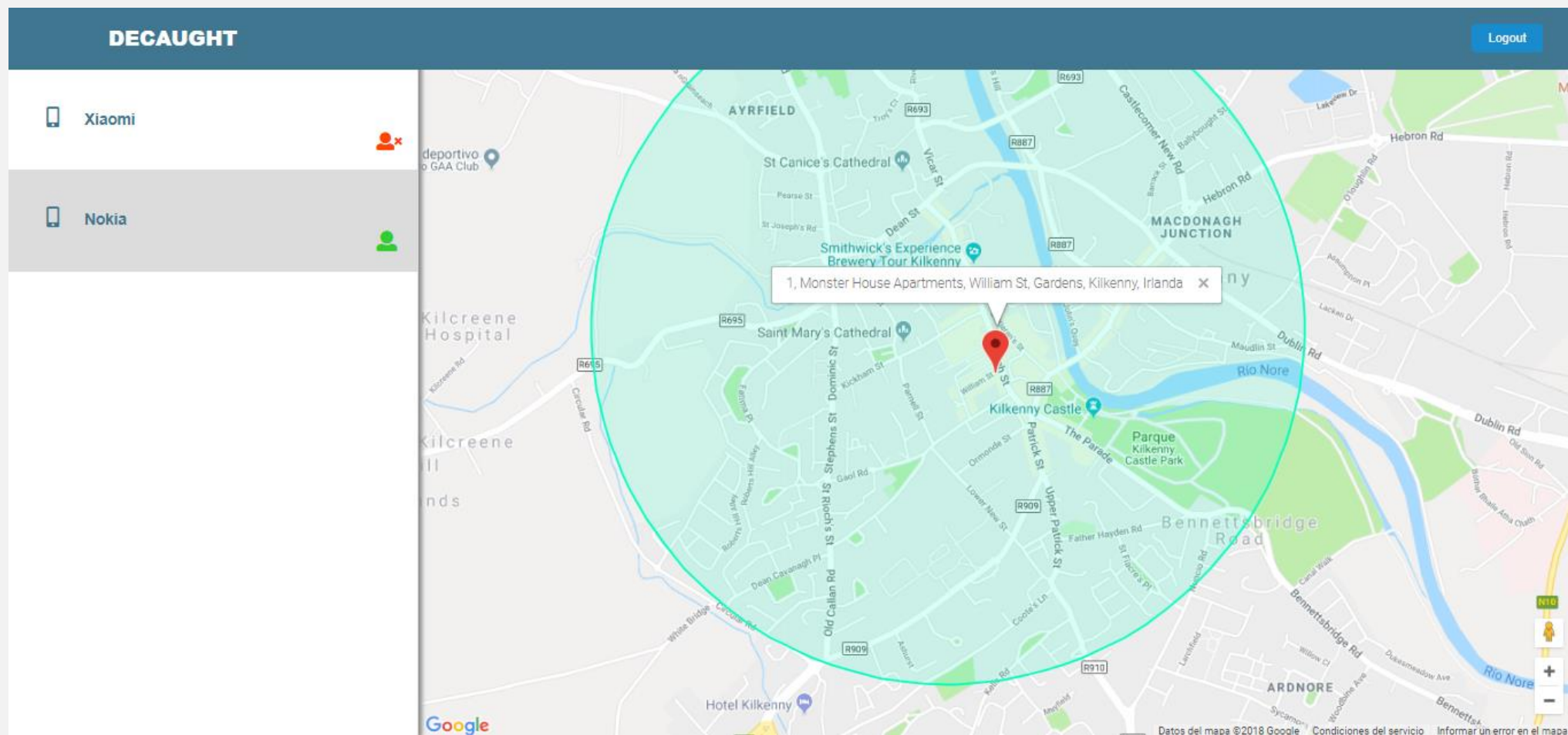
Password

[Forgot Password?](#)

 SIGN IN



# Página principal

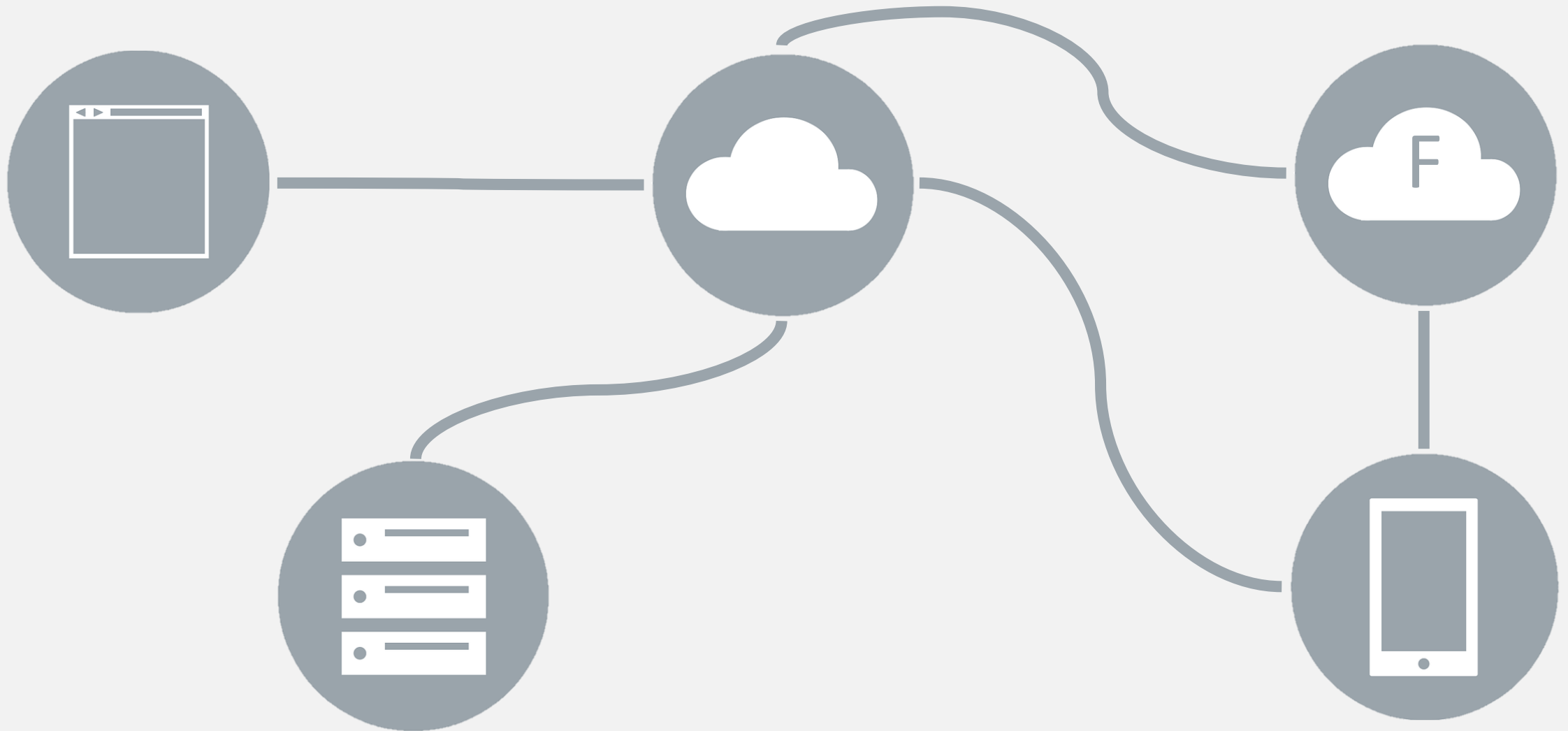


# Apartado Técnico

Visión general



# Diagrama de componentes



# Puntos clave del servidor

---



# Servidor

```
class Location(models.Model):
    device = models.ForeignKey(Device, null=False, on_delete=models.CASCADE, verbose_name='Dispositivo')
    data_time = models.DateTimeField(blank=False, null=False, verbose_name='Fecha y hora')
    latitude = models.DecimalField(max_digits=25, decimal_places=20, blank=False, null=False, verbose_name='Latitud')
    longitude = models.DecimalField(max_digits=25, decimal_places=20, blank=False, null=False, verbose_name='Longitud')
    im_out = models.BooleanField(default=False, verbose_name='Fuera')

    class Meta:
        verbose_name = 'Ubicación'
        verbose_name_plural = 'Ubicaciones'

def create(self, request, *args, **kwargs):
    try:
        device = Device.objects.get(user=request.user.pk, device_id=request.data['device'])
        data = {'device': device,
                'data_time': request.data['data_time'],
                'latitude': request.data['latitude'],
                'longitude': request.data['longitude'],
                'im_out': request.data['im_out']}
        serializer = LocationSerializer(data=data)
        if serializer.is_valid():
            serializer.save()
            response = Response(serializer.data)
            in_out = request.data['im_out']
            if in_out == "true":
                text = device.label + ' está fuera de las áreas seguras.'
                device = Device.objects.get(user=request.user, role='M')
                device.send_message("Alerta de ubicación", text, data={'device': request.data['device']})
            else:
                response = Response(serializer.errors)
    except Device.DoesNotExist:
        response = Response(status=status.HTTP_404_NOT_FOUND)
    return response
```

# Web

```
def login(request):  
    return render(request, 'login/index.html')
```

```
@login_required(login_url=settings.LOGIN_URL)
```

```
def main(request):
```

```
    devices = Device.objects.filter(user=request.user.pk, role='E')
```

```
    locations = []
```

```
    geofences = []
```

```
    for device in devices:
```

```
        locations.append(Location.objects.filter(device=device).order_by('-data_time').first())
```

```
        geofences.append(Geofence.objects.filter(device=device).first())
```

```
    context = {
```

```
        'device': devices,
```

```
        'location': locations,
```

```
        'geofence': geofences}
```

```
    return render(request, 'main/index.html', context)
```

```
urlpatterns = [
```

```
    url('login/', login, {'template_name': 'login/index.html'}, name='login'),
```

```
    url('main/', views.main, name='main'),
```

```
    url(r'^admin/', admin.site.urls),
```

```
    url(r'^$', include(router.urls)),
```

```
    url(r'^api-auth/', cosa.obtain_auth_token),
```

```
    url('reset/password_reset', password_reset,
```

```
        {'template_name': 'forget_pwd/password_reset_form.html',
```

```
        'email_template_name': 'forget_pwd/password_reset_email.html'},
```

```
        name='password_reset'),
```

```
    url('reset/password_reset_done', password_reset_done,
```

```
        {'template_name': 'forget_pwd/password_reset_done.html'},
```

```
        name='password_reset_done'),
```

```
    url('reset/(?P<uidb64>[0-9A-Za-z_-]+)/(?P<token>.+)/$', password_reset_confirm,
```

```
        {'template_name': 'forget_pwd/password_reset_confirm.html'},
```

```
        name='password_reset_confirm'),
```

```
    url('reset/done', password_reset_complete,
```

```
        {'template_name': 'forget_pwd/password_reset_complete.html'},
```

```
        name='password_reset_complete'),
```

```
]
```

# Puntos clave de la aplicación

---

¿Cómo la aplicación hace peticiones al servidor?

# Aplicación

```
private void signIn() {
    if (validate()) {
        mService.get_token(user.getText().toString(), password.getText().toString()).enqueue(new Callback<Token>() {
            @Override
            public void onResponse(Call<Token> call, Response<Token> response) {
                if (response.isSuccessful()) {
                    editor.putString("username", user.getText().toString());
                    editor.putString("password", password.getText().toString());
                    editor.putString("token", response.body().getToken());
                    editor.apply();
                    startActivity(new Intent(getApplicationContext(), RolSelectionActivity.class));
                    finish();
                } else {
                    int statusCode = response.code();

                    if (statusCode == 400) {
                        Toast.makeText(context, LoginActivity.this, "Usuario y/o contraseña incorrectos", Toast.LENGTH_SHORT).show();
                    }
                }
            }
        });

        @Override
        public void onFailure(Call<Token> call, Throwable t) {
            Toast.makeText(context, LoginActivity.this, "Error al conectar", Toast.LENGTH_SHORT).show();
            t.printStackTrace();
            Log.d("MainActivity", "error loading from API");
        }
    }
}
```

# Aplicación

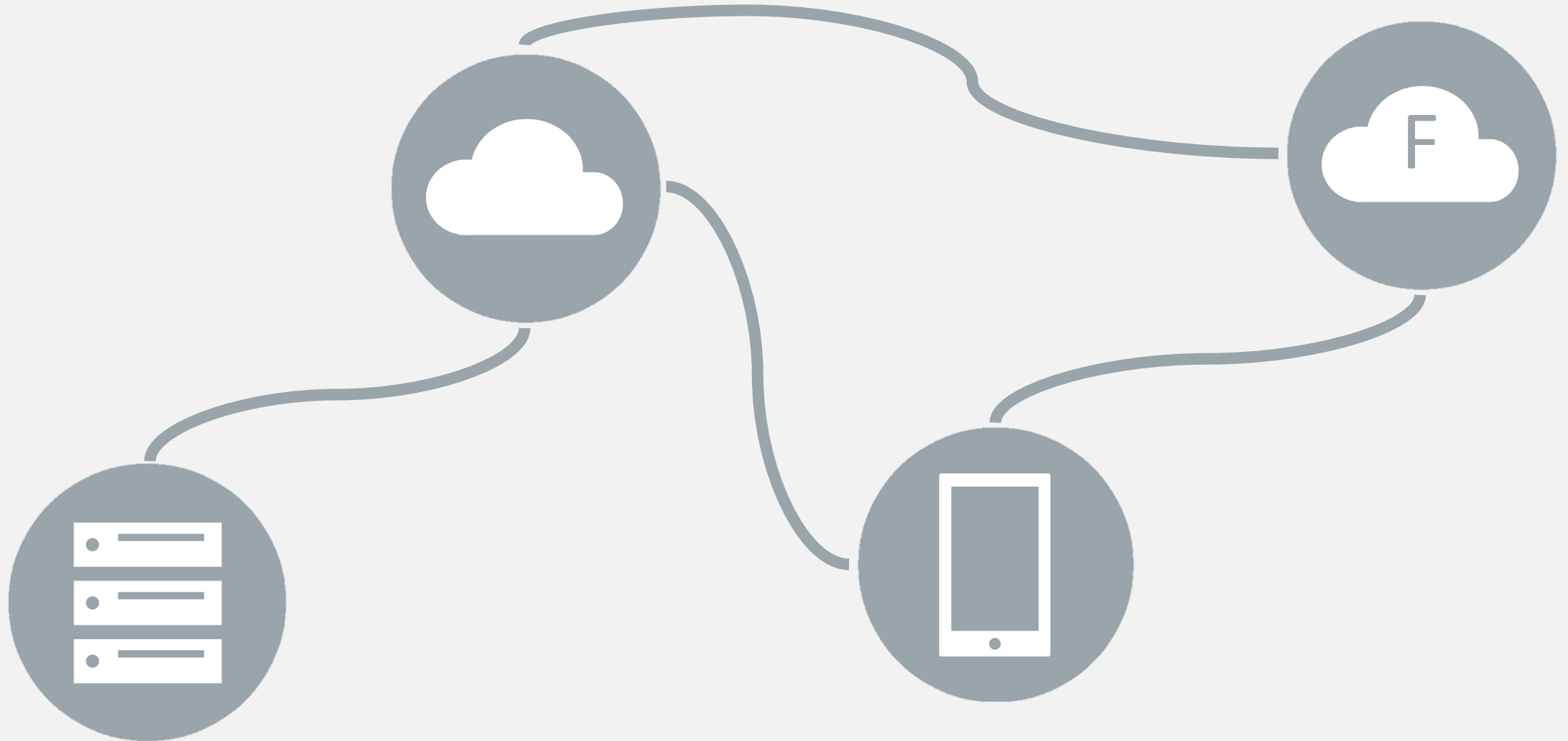
```
// Añadir contactos
@FormUrlEncoded
@POST("contacts/")
Call<Contact> add_contact(@Header("Authorization") String token, @Field("device") String device, @Field("phone") String phone, @Field("name") String name);

// Consultar contactos
@GET("contacts/")
Call<List<Contact>> get_contacts(@Header("Authorization") String token, @Query("role") String role);
```

```
public void processNotification(String title, String body, Map<String, String> data) {

    if (data.get("action") != null) {
        switch (data.get("action")) {
            case "update_geofences":
                downloadGeofences();
                break;
            default:
                break;
        }
    } else if (data.get("device") != null) {
        displayNotification(title, body, data.get("device"));
    } else {
        displayNotification(title, body, device: null);
    }
}
```

# Notificaciones



# Notificaciones

```
from fcm_django.models import FCMDevice

devices = FCMDevice.objects.all()

devices.send_message(title="Title", body="Message")
devices.send_message(title="Title", body="Message", data={"test": "test"})
devices.send_message(data={"test": "test"})
```



